Zadanie 2

Treść

Danych jest n odcinków $I_j = \langle p_j, k_j \rangle$, leżących na osi OX, $j = 1, ..., n$. Ułóż algorytm znajdujący zbiór $S \subseteq \{I_1, ..., I_n\}$, nieprzecinających się odcinków, o największej mocy.

Algorytm

Niech $I = \{I_1, ..., I_n\}$ będzie zbiorem danych odcinków.

1. Sortujemy odcinki z I niemalejąco według ich końców k_j. Niech posortowana lista odcinków to $I' = I'_1, ..., I'_n$.
2. Inicjalizujemy pusty zbiór wynikowy $S = \emptyset$.
3. Jeśli posortowana lista I' nie jest pusta:
   i. Dodajemy pierwszy odcinek do S, $S = \{I'_1\}$
   ii. Zapamiętujemy czas zakończenia ostatniego dodanego odcinka: `ostatni_koniec` $= k'_1$, gdzie k'_1 to koniec odcinka I'_1.
   iii. Dla każdego kolejnego odcinka I_j (gdzie $j = 2, ..., n$):
   - Jeśli początek odcinka I'_j jest nie mniejszy niż `ostatni_koniec` (tzn. $p'_j \geq$ `ostatni_koniec`):

     $S = S \cup \{I'_j\}$

     `ostatni_koniec` $= k'_j$
4. Zwróć S.

Dowód

Niech $\text{ALG} = a_1, ..., a_m$ będzie zbiorem odcinków wybranym przez nasz algorytm, posortowanym zgodnie z kolejnością wyboru (a więc również według niemalejących końców k_{a_j}). Niech $\text{OPT} = o_1, ..., o_k$ będzie dowolnym optymalnym zbiorem nieprzecinających się odcinków, o największej mocy k, również posortowanym według niemalejących końców k_{o_j}.

Chcemy pokazać, że $m = k$.

Rozpatrzmy następujące przypadki dotyczące relacji między końcami odcinków a_i oraz o_i:

1. $\boldsymbol{k_{o_i} < k_{a_i}}$:

   Ten przypadek prowadzi do sprzeczności. Gdyby $k_{o_i} < k_{a_i}$, to algorytm, wybierając odcinek o najwcześniejszym końcu spośród dostępnych (a o_i był dostępny i $p_{o_i} \geq k_{a_{i-1}}$), wybrałby o_i lub inny odcinek kończący się nie później niż o_i, a nie a_i, który kończy się później. Zatem ten przypadek jest niemożliwy. Musi być $k_{a_i} \leq k_{o_i}$. ↯

2. $\boldsymbol{k_{o_i} = k_{a_i}}$:

   Jeśli $a_i \neq o_i$ (bo to pierwszy różniący się element), ale $k_{a_i} = k_{o_i}$, możemy skonstruować nowe rozwiązanie $\text{OPT}' = \{o_1, ..., o_{i-1}, a_i, o_{i+1}, ..., o_k\}$. Rozwiązanie OPT' jest poprawne:

   - a_i nie koliduje z o_{i-1} (ponieważ $a_{i-1} = o_{i-1}$ i z definicji algorytmu $p_{a_i} \geq k_{a_{i-1}}$).
   - a_i nie koliduje z o_{i+1} (ponieważ $p_{o_{i+1}} > k_{o_i}$, a $k_{o_i} = k_{a_i}$, więc $p_{o_{i+1}} \geq k_{a_i}$). Liczba odcinków w OPT' jest taka sama jak w OPT (czyli k), więc OPT' jest również optymalne. Co więcej, OPT' zgadza się z ALG na co najmniej i pierwszych pozycjach. W tej sytuacji wybór a_i przez algorytm jest co najmniej tak dobry jak wybór o_i.

Opracował: Filip Figzał, 2025

3. $\boldsymbol{k_{o_i} > k_{a_i}}$:

   Podobnie jak w przypadku 3, skonstruujmy $\text{OPT}' = (o_1, ..., o_{i-1}, a_i, o_{i+1}, ..., o_k)$. Rozwiązanie OPT' jest poprawne:

   - a_i nie koliduje z o_{i-1}.
     - $p_{o_{i+1}} gek_{o_i}$. Ponieważ $k_{o_i} > k_{a_i}$, to tym bardziej $p_{o_{i+1}} > k_{a_i}$ (a więc $p_{o_{i+1}} gek_{a_i}$). Zatem a_i nie koliduje z o_{i+1}.

   Liczba odcinków w OPT' wynosi k, więc OPT' jest optymalne. OPT' zgadza się z ALG na co najmniej i pierwszych pozycjach. Wybór a_i (który kończy się wcześniej) jest co najmniej tak samo dobry (a potencjalnie lepszy, bo zostawia więcej miejsca) jak wybór o_i.

W każdym możliwym przypadku, możemy zmodyfikować OPT tak, aby zgadzało się z ALG na i-tej pozycji, nie tracąc optymalności ani poprawności. Powtarzając ten argument dla kolejnych pozycji $j > i$, na których ALG i (zmodyfikowane) OPT mogłyby się różnić, możemy krok po kroku przekształcić całe OPT w ALG, nie zmniejszając liczby odcinków. Oznacza to, że liczba odcinków w ALG jest co najmniej tak duża jak w OPT, czyli $m \geq k$. Ponieważ z definicji OPT jest rozwiązaniem optymalnym, $m \leq k$. Łącząc te dwie nierówności, dochodzimy do wniosku, że $m = k$. Zatem algorytm jest poprawny i zawsze znajduje rozwiązanie o największej mocy